DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Primer Trabajo Práctico

14 de Abril de 2010 — Algoritmos y Estructuras de Datos III

Integrante	LU	Correo electrónico
Bianchi, Mariano	92/08	bianchi-mariano@hotmail.com
Brusco, Pablo	527/08	pablo.brusco@gmail.com
Di Pietro, Carlos Augusto Lyon	126/08	cdipietro@dc.uba.ar

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Ejercicio 1

1.1. Introducción

El primer problema del presente trabajo consistió en la implementación de un algoritmo capaz de dar solución a la ecuación

$$b^n \ mod \ (n) \tag{1}$$

haciendo uso de alguna de las técnicas algorítmicas aprendidas hasta el momento en la materia. Asímismo, la consigna dictaba que la complejidad final del algoritmo debería ser menor a $O(n)$. En pos de cumplimentar lo pedido se decidió usar la técnica de *Dividir & Conquistar*[1] para desarrollar el algoritmo. Esta técnica se caracteriza principalmente en dividir la instancia de un problema en instancias más pequeñas, atacar cada una de ellas por separado y resolverlas, para finalmente juntar sus resultados y así producir el resultado final.

1.2. Explicación

La primera solución que se piensa casi de manera intuitiva es la de mutliplicar n veces el número b y luego hallar el resto de dividir ese resultado por n.

$$\underbrace{b.b.b \ldots b.b.b}_{n \ veces} \ mod \ (n) = b^n \ mod \ (n) \tag{2}$$

Sin embargo, la complejidad ese algoritmo es $O(n)$, ya que se realizan n multipicaciones y 1 división, razón por lo cual no cumple con lo pedido en la consigna. Además, puesto que ese algoritmo realizar sucesivas multipliciones de b, cabe la posibilidad de que el valor en que se va acumulando el resultado sobrepase el máximo número entero representable en la computadora produciedo así un overflow y obteniéndose en consecuencia un resultado incorrecto.

Analizando la falla del algoritmo anterior, se observa que lo que se debe evitar es calcular directamente el resultado de b^n ya que ese número podría ser excesivamente grande.
Con esto en mente veamos una manera más conveniente de abordar el problema. Es claro, por definición de congruencia, que resover la ecuación en 1 equivale a hallar el x tal que:

$$b^n \ \equiv \ x \ (n) \ , \quad con \ 0 \ \leq \ x \ < \ n \tag{3}$$

Luego, asumamos por un momento que podemos tener gratis un k tal que:

$$b^{n/2} \ \equiv \ k \ (n) \ , \quad con \ 0 \ \leq \ k \ < \ n \tag{4}$$

Entonces, aplicando el resultado de 4 en 3 obtenemos que:

- Si n es par:

$$\begin{aligned} b^n \ \equiv \ x \ (n) \ , \quad con \ 0 \ \leq \ x \ < \ n \\ b^{n/2}.b^{n/2} \ &\equiv \ x \ (n) \\ k.k \ &\equiv \ x \ (n) \\ k^2 \ &\equiv \ x \ (n) \end{aligned} \tag{5}$$

[1] Poner alguna referencia en donde se explique esta técnica

- Si n es impar par:

$$\begin{aligned} b^n &\equiv x\ (n)\ , \quad con\ 0\ \leq\ x\ <\ n \\ b.b^{n/2}.b^{n/2} &\equiv x\ (n) \\ b.k.k &\equiv x\ (n) \\ b.k^2 &\equiv x\ (n) \end{aligned} \tag{6}$$

Sin lugar a dudas, las expresiones resultantes en ambos casos son fáciles de resolver puesto que números mucho menores que b^n.

2. Ejercicio 2

2.1. Introducción

El segundo problema consistió en la implementación de un algoritmo capaz de dar solución al problema que se plantea a continuación:

Decidir si un grupo de n personas pueden formar o no una ronda que cumpla con las siguientes restricciones:

- La ronda debe contener a todas las personas.
- Algunas personas son amigas y otras no.
- Cada alumna debe tomar de la mano a dos de sus amigas.

2.2. Explicación

Dado que para este problema no se conocen algoritmos buenos[2], se pensó en utilizar la solución por fuerza bruta, con algunas mejoras, es decir, intentar todas las combinaciones hasta lograr determinar si hay o no solución.

A este método, se le agregó

[2]Un algoritmo se considera bueno si puede ser resuelto en tiempo polinomial.